CDIO Project - Images and Graphics

Robotics Safety

Olle Fridolfsson
Niklas Hansson
Patrik Hillgren
Benjamin Ingberg
Pär Lundgren
Mattias Nilsson

December 11, 2013

Preamble

We would like to thank Johan Hedborg and Rickard Olssen for guidance and support throughout the project.

Contents

1 Background

This is the technical documentation of the work done by the Robotic Safety group in the course TSBB11 at Linköping University. TSBB11 is a CDIO project course given by the Computer Vision Laboratory where students apply their knowledge from previous theoretical courses in a larger project. The members of the group Robotic Safety all study the Signal- and Image Processing master program. The project assigned to the group is in the area of robotics safety and is requested by IEI (Institutionen för industriell och ekonomisk utveckling/Department of Management and Engineering) at Linköping University and relates to the robot SiA20 Motoman by Yaskawa nordic.

2 Project Description

Robots like SiA20 Motoman are powerful and can in some situations cause harm to humans working in the same area. The goal of this project is to create a system which ensures a safe environment in a human robot collaboration area. This will be done by determining the distance between the closest moving object and the robot and retrieve a control signal to send to the robot depending on that information. The control signal should determine four states:

1. The closest moving object is outside safety zone 2 (see figure below). Objects are not in the collaborative area, the robot can work at standard motion.
2. The closest moving object is within safety zone 2. This means that the robot motion shall be reduced since the moving object is within the collaborative area.
3. The closest moving object is within safety zone 1. This means that the robot must stop.
4. The closest moving object is within the emergency zone. This means that the robot must perform an emergency stop, which differs from the usual stop.

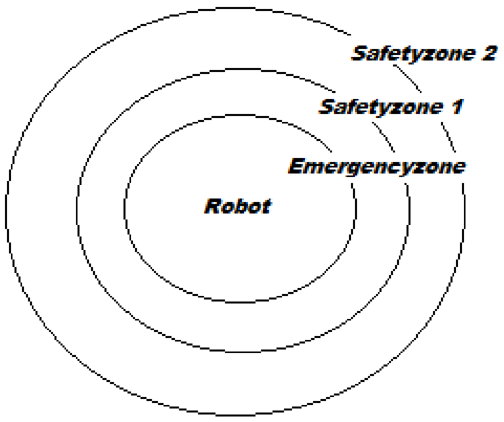

Figure 1: Flowchart of tracking algorithm

3 Development Environment

3.1 Robotic Operating System (ROS)

The implementation is done in ROS-packages according to ROS-standards. ROS uses sockets (topics) to send data between different programs (nodes). The nodes and topics are very useful since it gives the opportunity to run the system partitioned in different programs. The main advantages of using ROS is the possibility of simple communication with the robot. ROS also provides information about the whereabouts of the robot which in this system is of high interest. Another good thing about ROS is that it contains many third party libraries, common for projects including robots and computer vision, such as OpenCV and Point Cloud Library.

3.2 External libraties

OpenCV, Point Cloud Library(PCL).

4 System Overwiev

For monitoring the area surrounding the robot a Microsoft Kinect is used. Together with data that is received from the controller it is possible to decide whether a visible object is within a safe distance from the robot, or not. The data from each source is transformed into a joint environment.

The system consist of:

- A Kinect - a motion sensing input device by Microsoft.
- Yaskawa DX100 controller - provided with a MotoPlus application.
- Yaskawa SIA20 robot - A slim version of it predecessor IA20.
- A Computer running the system.

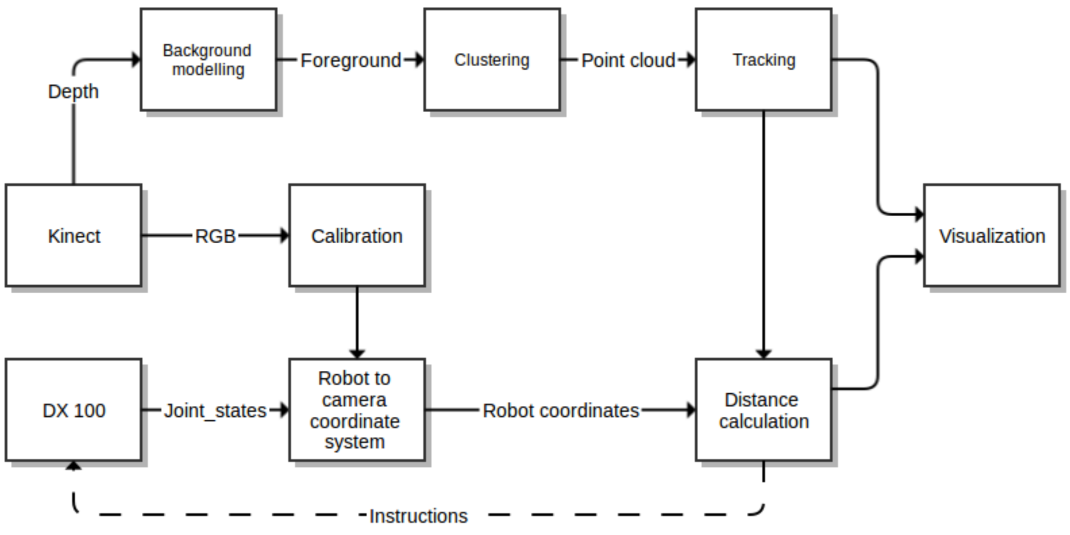

Figure 2: Flowchart of system

From the Kinect a depth image is provided. A background model algorithm is applied on the depth image and a foreground is obtained. The foreground only consist of moving pixels and is fairly rough. Therefore a clustering algorithm is used to extract only the relevant data. Clusters are easier to handle and much easier to track. Tracking provides a backup for what the background model might fail on. E.g. since objects might be left out from the foreground if they stay at the same place for too long.

The Kinect also provides an RGB image which makes it possible to perform calibration of the camera extrinsics using calibration patterns. Calibration algorithms are used to find transformations between different coordinates systems. This is crucial in order to measure the distance between the robot and an object.

The visualization displays a point cloud of the objects together with the robot model. Lines are used to represent the closest point pairs. The color of the line indicates in which zone the object is situated. Red emergency zone, yellow safety zone 1 and green safety zone 2.

The original idea was to send instructions to the controller. However integrating the robot in the system ended up being very time consuming. Since the topic of this project is computer vision and image processing the group decided to leave the instruction sending part out. It is possible to further develop the project to include robot control based on decisions made by our system.

5 Subsystems

This section describes in detail the different parts which are included in the system.

5.1 Input Data

The different types of input data to the system.

5.1.1 Joint states of robot

When connection with the controller is established the joint states will be available. Each joint has a value in radians describing its state, i.e. its revolution. The objective is to build a model of the robot in camera coordinates where the distance to a moving object, represented as a point cloud in 3D space can be computed. The data regarding joint states is used for both visualization, in shape of a mesh figure, and for calculation of the robot position in relation to other objects. These two are independent with respect to each other.

5.1.2 Depth-image from Kinect

The IR projector and camera provides a depth image of the collaboration area. The data is used for background segmentation and further on for clustering and tracking. Calibration of the IR cameras intrinsic parameters is performed as well. This ensures a correct transformation between world and camera coordinates.

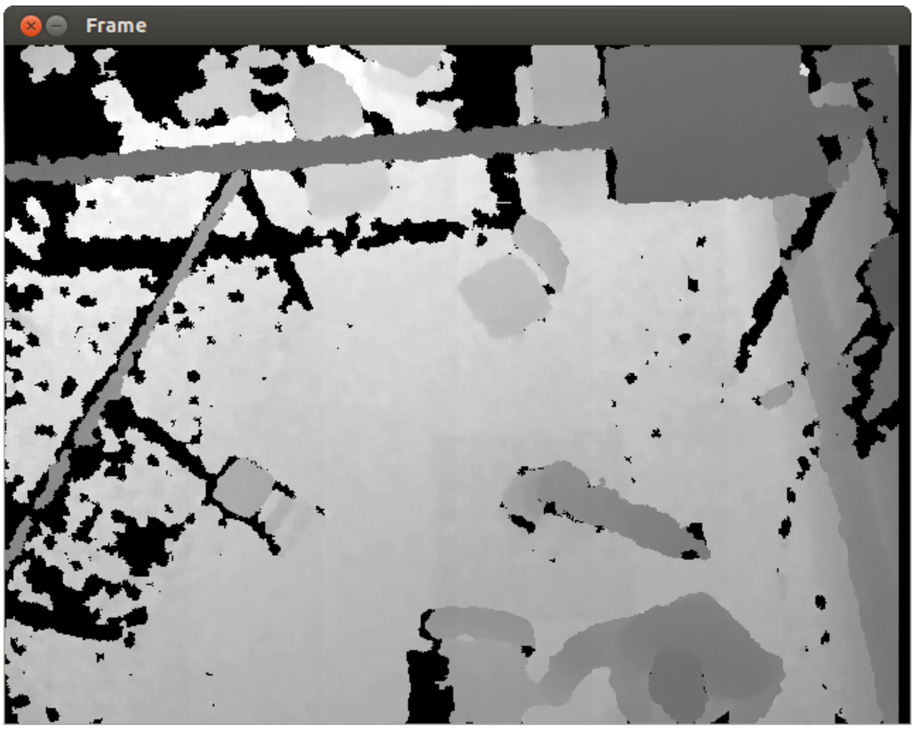

Figure 3: Depth image

5.1.3 RGB-image from Kinect

The RGB camera is used for calibration of both intrinsic and extrinsic camera parameters. From those it is possible to determine a relationship between the robot and a well known positioned calibration pattern used in calibration.

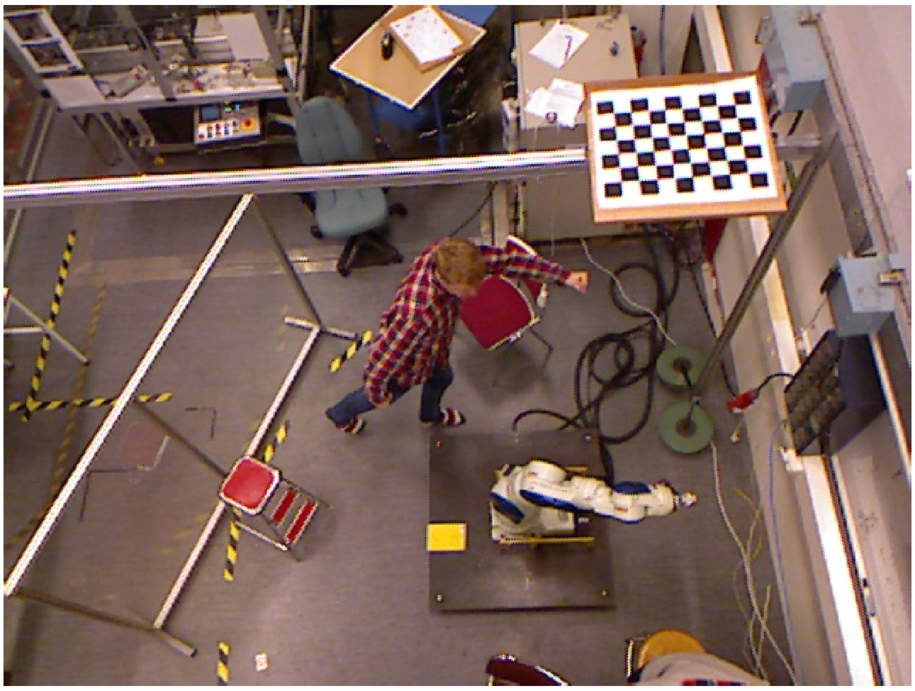

Figure 4: RGB-image

5.1.4 tf

The tf subsystem is a transformation management system written and maintained by Tully Foote. Since the system will have dozens of frames the tf subsystem traverser the set of known frame relations to give the transformation parameters between two arbitrary reference frames at any point in time.

5.2 Calibration

To relate the camera system to the world system the system has to be calibrated. For calibration of the camera intrinsic parameters and distortions from the pinhole camera model the system uses the models and methodology described by Zhang, 2000 [4]. This is performed offline.

For online calibration of the camera extrinsics the same distortion parameters are used with a world fixed calibration pattern.

The online calibration is very computationally expensive however since the system is supposed to be stationary recalibration is only done once a second to compensate for small adjustments on the system.

Most of the building blocks of the calibration is implemented by the OpenCV library [5].

5.2.1 Intrinsic calibration

Before image distortions the projection of the 3D-world to the image plane is described by this matrix:

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Where f_x and f_y are the number of pixels per unit of length, cx and cy are the center pixel coordinates of the image. X, Y and Z are the camera relative positions of a visible point with x, y and w as the homogeneous representation of the pixel coordinate where the point is projected.

Distortions from this model is modelled as radial distortion as well as tangential distortions. Radial distortions produce a fish-eye effect on the image and is compensated for with this model:

$radial_corrected_x = f(x)$ $radial_correcced_y = f(y)$

Tangential distortions due to the lense not being perfectly aligned is compensated with this model:

$tangential_corrected_x = g(x)$ $tangential_corrected_y = g(y)$

Since this is a standard model external libraries have support for these parameters and the error correction from calculating the distortion parameters will cascade into other functions. Intrinsic and distortion calibration is done offline and the parameters are saved into a YAML file.

5.2.2 Extrinsic calibration

To relate the position and state of the robot to the camera there needs to be a reference between a fixed robot frame and the camera frame. To solve this a calibration pattern is placed at a known position relative to the robot.

This calibration pattern is then detected in the image and the solution to the PnP (Perspective-n-Point) problem with the camera parameters is used as extrinsic parameters.

To minimize noise and increase robustness a large chessboard pattern with 6x8 known points is used for calibration. However there is no known analytical solutions to the PnP problem for n $>$ 3 so the problem is solved using optimization (minimization of the sum of squared reprojection errors) with the Levenberg-Marquardt iterative optimization algorithm.

To prevent the optimization getting stuck in local optima the optimization is done on an analytical solution to the P3P problem. The points used for the analytical solution are the four corners of the chessboard, three to solve the P3P problem and one to validate which of the four possible solutions is consistent with the rest of the data.

The solution is then further median filtered on one of the rotation parameters to prevent outliers. Though running the system has so far yet to produce an outlier after the previous errors these median filtered points could in the future be averaged to further increase precision if the system demands it.

5.2.3 Transformation of robot joints

While running warehouse_viewer the the transformations between the joints can be obtain given the relation of the joints specified in the robot's URDF file, see appendix. The tf package included in ROS makes it possible to transform the robot and its joints into a given coordinate system, provided by the RGB camera.

5.2.4 Transformation into camera coordinate system

A static transformation between calibration pattern and the base of the robot in world coordinates is set. Given the transformation from camera to world coordinates it is possible to transform each joint to the camera coordinate system. This results in a set of joints on which the distance calculation to moving objects in the scene can be performed.

5.3 Background segmentation

The background segmentation node is based on the backgroundSubtractorMOG2 from OpenCV [1] which uses Gaussian mixtures to perform the segmentation. The background segmentation results in a binary image of the foreground.

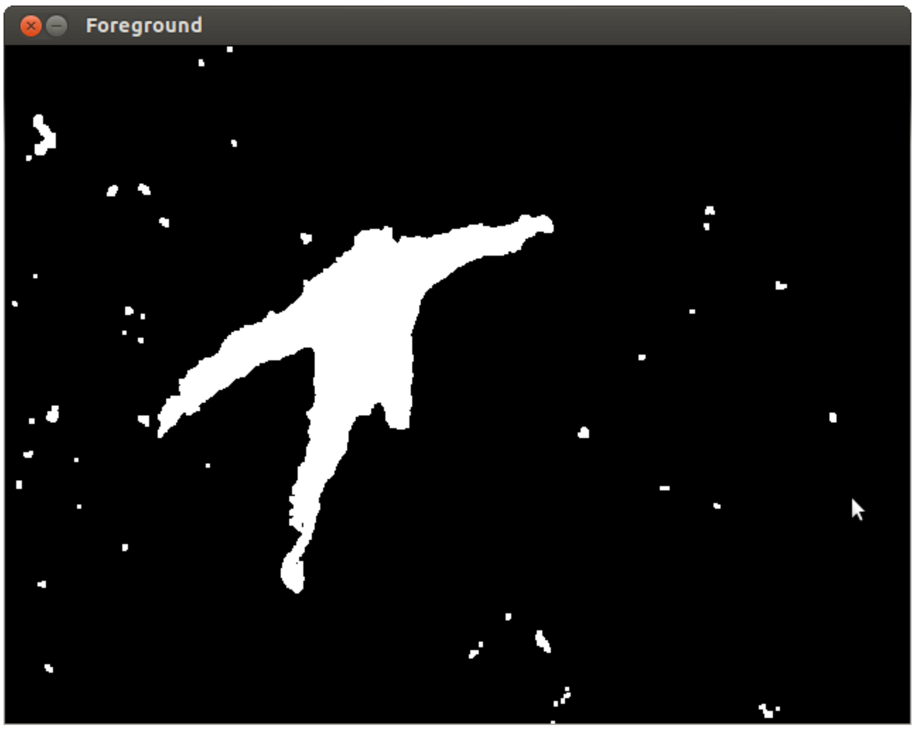

Figure 5: Foreground from background segmentation

The foreground and depth image are used to construct a point cloud. For each foreground pixel the corresponding depth value P_z, pixel coordinates P_x, P_y and intrinsic parameter can be used to obtain the world coordinates as follows.

$$P_x = (p_x - c_x) \cdot \frac{P_z}{f_x}$$

$$P_y = (p_y - c_y) \cdot \frac{P_z}{f_y}$$

and the point $\mathbf{P} = [P_x, P_y, P_Z]^T$

and the point $P = [P_x, P_y, P_Z]^T$

The foreground is converted to a point cloud that describes the shape of the foreground in camera coordinates. Using the foreground and depth image it is possible to construct a point cloud. For each foreground pixel the corresponding depth value P_z, pixel coordinates p_x, p_y and intrinsic parameters can be used to obtain the world coordinates as follows.

By applying these formulas to all foreground pixels the point cloud is achieved.

5.4 Clustering

The clustering node receives a point cloud from the background segmentation node corresponding to the extracted foreground i.e. moving objects. The purpose of the clustering node is to divide the entire point cloud into individual objects. It also removes noise and objects that are not considered to be large enough.

The reason why individual objects are necessary is because the closest distance to every object compared to the robot is desired. It is not sufficient to know the distance from the closest point in the whole point cloud to the robot. Separating the entire point cloud into individual clouds also provides the possibility of tracking point cloud objects.

The separation of objects is done by separating the entire point cloud into clusters. This is done using an euclidean cluster extraction algorithm. A simple outline of the algorithm:

```
Data: Point cloud, Cluster parameter
Result: Clustering
initialization;
for every point not in a cluster do
    for every other point do
        check distance to point;
        if distance < Cluster parameter then
            assign to the same cluster;
        else
            assign to a new cluster;
        end
    end
end
```

Algorithm 1: How to write algorithms

1. for each point
2. check distance to all other points
3. if a point is closer than parameter
4. assign this point to the same cluster
5. if no matching point is found
6. assign to a new cluster

The cluster extraction is based on the algorithm given by Point Cloud Library [2].

5.5 Removal of Robot

In the background segmentation the system finds everything that is classified as foreground, including the robot. This results in a point cloud that might contain values corresponding to the robot. The purpose of this project is to check if detected objects are within a certain distance from the robot. If the robot itself is one of the objects the distance will always be minimal and the system will not work. This means that the system needs to remove clusters corresponding to the robot from the other clusters before it can calculate distances and make a decision.

Removing the clusters corresponding to the robot is done by using the joint coordinates of the robot. Since the system only knows the joints position and not the entire hull of the robot the first thing that needs to be done is to create a convex hull corresponding to an approximation of the real hull of the robot. The approximated hull is achieved by creating lines between connected joints and applying a cylinder around these lines. The radius of the cylinder is given by the greatest radius of the robot between these joints. In other words the cylinders corresponding to the hull of the robot will cover up the whole robot. After the hull is created a simple check is performed which finds out if a certain amount of the points in a cluster are inside any of these cylinders. If this is the case these clusters are excluded for further processing. It is not trivial to know if a point is inside a cylinder, the used method is copied from Greg James. [3] The pseudo code for the algorithm is described below.

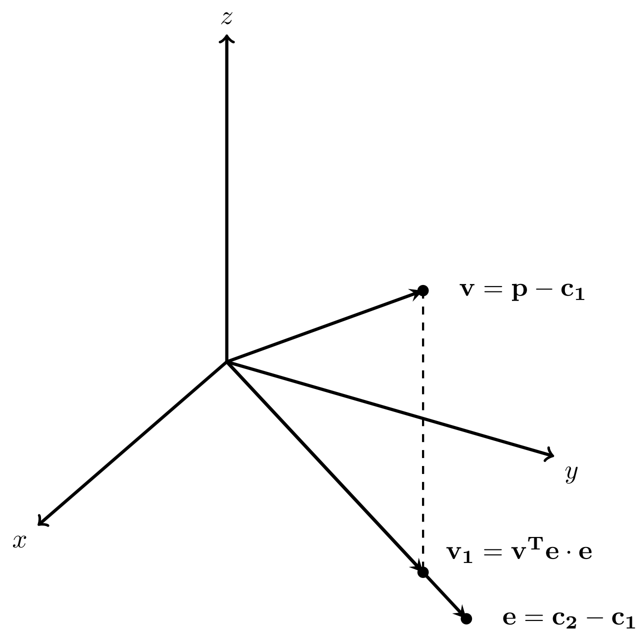
z
$\mathbf{v} = \mathbf{p} - \mathbf{c_1}$
y
x
$\mathbf{v_1} = \mathbf{v^T e} \cdot \mathbf{e}$
$\mathbf{e} = \mathbf{c_2} - \mathbf{c_1}$

Figure 6: Geometry for computing the orthogonal distance between two vectors

The objective is to conclude if $\mathbf{p}$ is inside a cylinder with squared radius r^2 and endpoints $\mathbf{c_1}$, $\mathbf{c_2}$. First $\mathbf{c_1}$ is subtracted from all points to get the vectors shown in figure 6. $\mathbf{v_1}$ is obtained by projecting $\mathbf{v}$ onto $\mathbf{e}$.

1. If the vector $\mathbf{v}$ lies between the caps (endpoints) of the cylinder, the

following is fullfilled: $0 < \mathbf{v}^T\mathbf{e} < \|\mathbf{e}\|^2$ if this is not the case, this distance is discarded since it is closer to another cylinder.

«««< HEAD

2. The orthogonal squared distance using Pythagoras formula is:

=======

3. The orthogonal squared distance using pythagoras formula is:
»»»> 2ba0b57d67dfac4b2eb97de65bc150a08125447b

$$d^2 = \|\mathbf{v}\|^2 - \|\mathbf{v_1}\|^2 = \mathbf{v}^T\mathbf{v} - \frac{(\mathbf{v}^T\mathbf{e})^2}{\mathbf{e}^T\mathbf{e}}.$$

4. if $d^2 < r^2$ the point is inside the cylinder.

This procedure is carried out for all points in each cluster for all cylinders spanning the robot. The number of points inside a cylinder for each cluster is used to determine if the cluster corresponds to the robot. If the number is above a certain threshold, I.E 80% the cluster is classified as the robot and discarded.

5.6 Distance Calculation

For each cluster of points extracted, the remaining problem is to find the smallest distance to the robot. This is done by brute force, the system iterates through all clusters and for each point it finds the closest euclidean distance to the robot. The data which is saved is the distance itself, the point in the cluster and the joint of the robot which corresponds to the closest distance. The reason for saving the closest point in the cluster and the robot joint is to make it possible to visualize the closest point-joint pair. To gain performance the distance calculation is performed simultaneously as the system is removing parts that correspond to the robot. That way looping through all points several times in a row is avoided.

6 Tracking

The system is using a background segmentation which have a certain learning rate, this means that the system constantly learns what is considered background during runtime. The background modelling will however not be able to handle all kinds of situations. There is, for example, a theoretical possibility for a person to walk into the scene, stand still for a sufficient period of time and blend into the background. This problem is solved by the use of a tracker which tracks objects found in the scene. The tracking algorithm will save information of objects that disappears in the scene and the robot will not continue working in normal pace until the object has left the safety zone. The system is not required to keep track of objects that is occluded and objects that interfere with each other. Since the requirement of the tracker is rather low the implementation is simple and has no guarantees of actually tracking objects throughout the entire scene if occlusion occurs.

To keep track of all objects in the scene an object list is used containing a vector of all objects and an index towards the object which is closest to the robot. Every object contains the closest joint on the robot, closest point on the object and the distance between them, which is called the min-distance. It will also contain an average point and a bool-variable which tells if the object is visible or not. Every cluster which is classified as an object will then correspond to an object in the objectlist. When an object is calculated in a new frame its average point will be compared to the average points of the objects in the objectlist. If there are any objects in the objectlist which doesn't correspond to any objects in the current frame there are theoretically two different possibilities, the object has moved out of the camera view or the object has blended into the background. To know which one it is the min-distance of the object is used. If the distance is sufficiently large it is assumed that the object has moved out of the camera view and the object will then be removed from the objectlist. If the distance is sufficiently small the object is assumed to have blended into the background, the object will then be kept in the objectlist but it will be marked as not visible. Objects that are outside the safety zones are not really interesting and will not be tracked. Since new objects can't occur inside the safety zones another advantage with the tracking is that it will reduce noise.

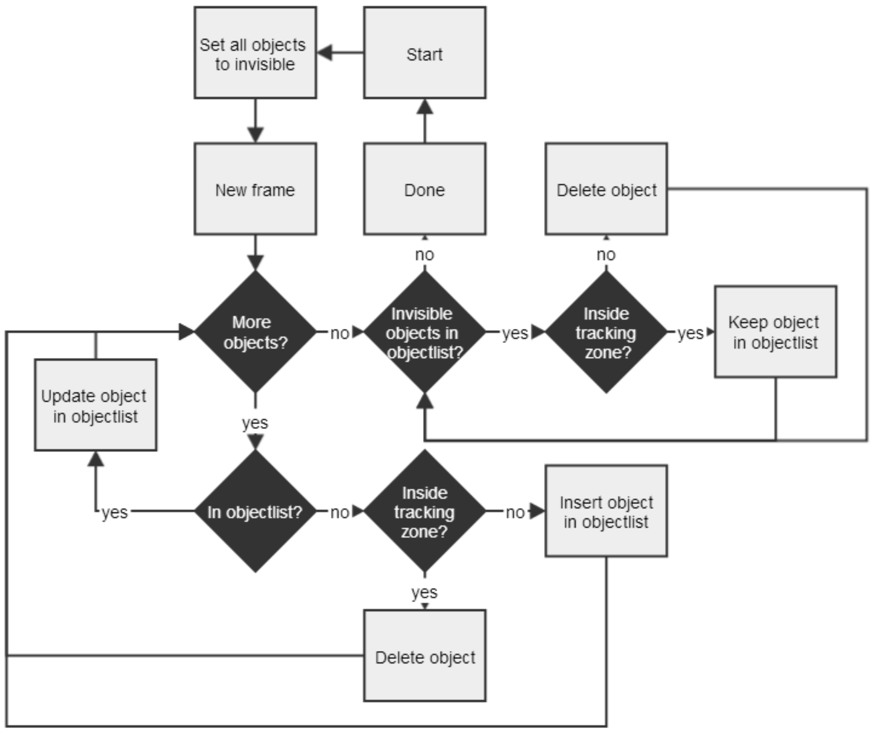
Set all objects to invisible
Start
New frame
Done
Delete object
no
no
More objects?
no
Invisible objects in objectlist?
yes
Inside tracking zone?
yes
Keep object in objectlist
Update object in objectlist
yes
yes
In objectlist?
no
Inside tracking zone?
no
Insert object in objectlist
yes
Delete object

Figure 7: Flowchart of tracking algorithm

6.1 Visualization

The results of the system is visualized in rviz, which is a 3D visualization tool for ROS.

6.1.1 Robot

The data published by the controller in JOINT_STATES are the radial position of the joints. The warehouse_viewer will use these states transformed into the coor The URDF file specifies the links between the robot's joints and the mesh files (.stl) and tf will contain the transformations between the joints which are used in warehouse_viewer to create 3D model of the robot.

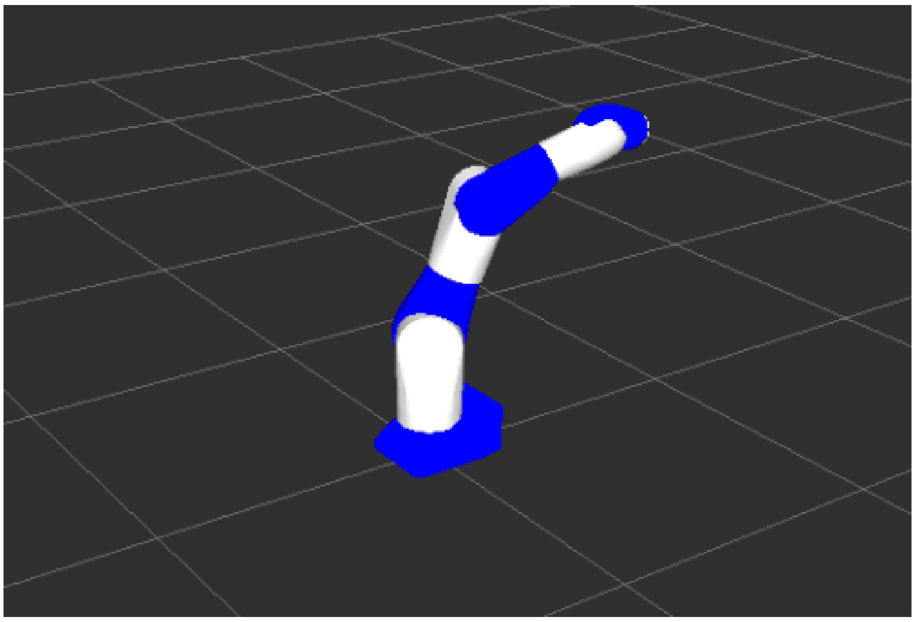

Figure 8: 3D-model of robot.

6.1.2 Objects

The objects in the image that the tracking algorithm considers being real moving objects are visualized. They are published as point clouds so that rviz can subscribe them and visualize them. If the tracking algorithm believes that an object has disappeared because of non-movement, it publishes the last seen point cloud of the object i.e. static objects in the scene are objects that the system thinks are standing still.

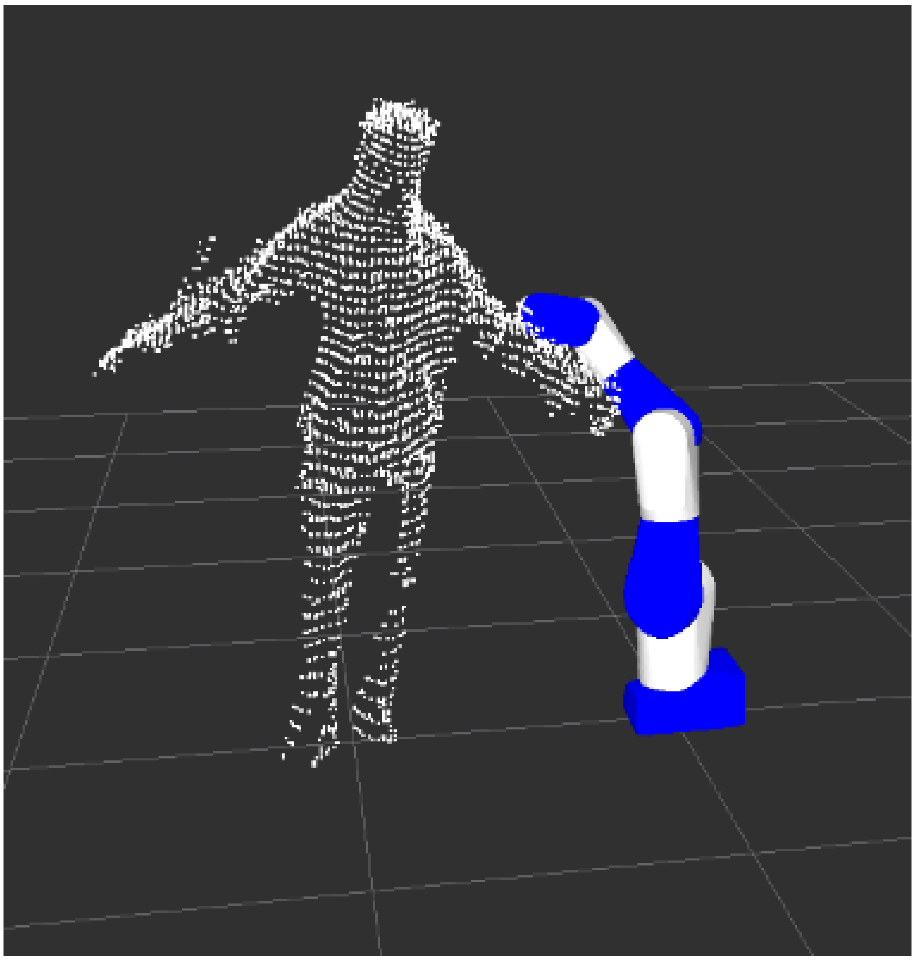

Figure 9: Human beside the robot.

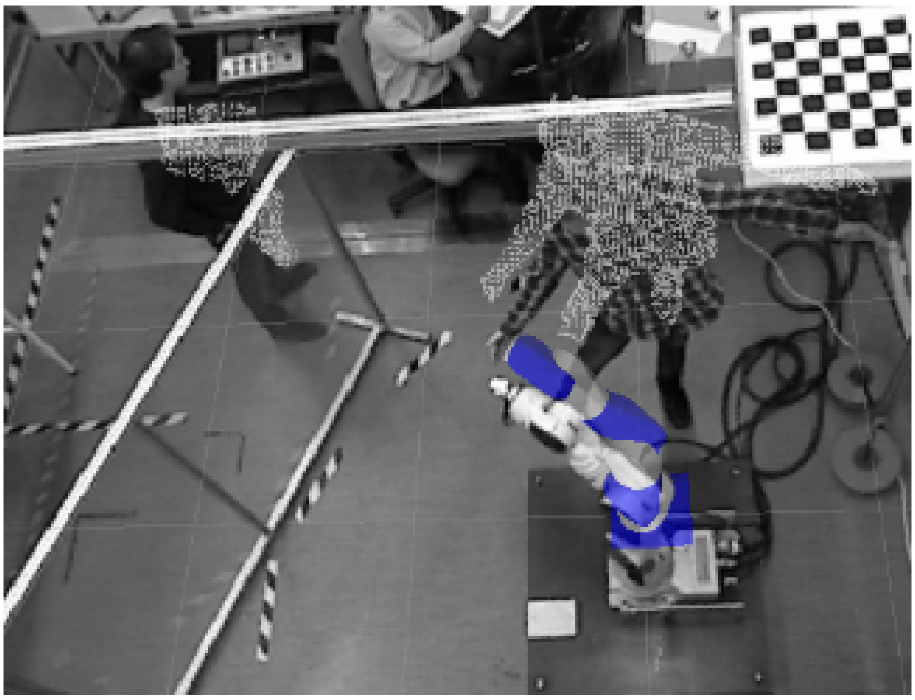

Figure 10: Humans and robot placed in world coordinate system.

6.1.3 Distances

To visualize the resulting state of the system (the current safety zone) a line between the closest object and the closest joint of the robot visualized. This line switches color depending on which state the system is in. The color of the line is thereby the result of the system.

- Red - Emergency zone
- Yellow - Safety zone 1
- Green - Safety zone 2
- No line - Outside safety zone 2

//lägg in en bild på hela systemet

7 Results and Evaluation

In this type of project there is no typical solution which final results can be evaluated by comparing the results with ground truth values. This is simply because there is no ground truth available for this type of problem. Evaluation of the system must then be done by checking if the system works for all possible scenarios. This type of evaluation is not quite desirable since it will not give a result which can be compared to other implementations of the same or similar problem. However, it will give a good understanding if the system works or not and motivates if the system could be used in real life collaboration areas between robots and humans. The following scenarios have been evaluated:

1. A person walks in into the scene, enters the outer security zone, stops and then leaves. This is the most trivial test and will show if the system produces any valuable results. This situation should be tested from all directions and the test person should enter all zones.

2. A person walks in into the scene, enters the outer security zone, stops for a long period of time and then leaves. This situation should be tested from all directions and the test persons should enter all security zones.

3. Two persons enter the scenes, enters the outer security zone, stops for a short period of time and then leaves. This situation should be tested from all directions and the test persons should enter all security zones. The path of the two persons should not interfere with each other, that is keeping a clear distance from each other.

4. Two persons enter the scenes, enters the outer security zone, stops for a short period of time and then leaves. This situation should be tested from all directions and the test persons should enter all security zones. The path of the two persons should interfere with each other, that is at one point keep a very small distance between each other.

5. A person enters the scene, enters the outer security zone, stops for a short period of time and then leaves. Before the person leaves it leaves an object behind within the outer security zone.

6. Two persons enter the scene maintaining physical contact (enough for them to appear as one object in the cluster extraction), enters the outer security zone, terminates the physical contact (the cluster extraction should now extract two objects) and then both leave the scene.

7. Two persons enter the scene maintaining physical contact (enough for them to appear as one object in the cluster extraction), enters the outer security zone, terminates the physical contact (the cluster extraction should now extract two objects) and then one person leaves the scene while the other stays in the outer security zone.

8. Multiple persons enters the scene acting like a herd of sheeps (moving randomly without knowledge about the robot). The persons should once in a while enter security zones. This test makes it possible to count the number of correct classifications and this test more or less correspond to a real life situation.

7.1 Performance of system using tracking functionality

The tracking algorithm performs very well when a single object is within the collaborating area. The tracker stays with the object and holds on to it, even when it is no longer visible in the foreground.

A person can enter the area, have its distance to the robot visualized with either a green, yellow or red marker. The marker is illustrated in form of a cylinder between the robot and the object. The marker might switch between two joint that are approximately on the same distance or between the knees of a person.

When two or more objects are within the collaborating area the system gets choppy. The current computer does not manage to process all data in real time and this produces bugs. When two or more objects are within the collaborating area there is a risk that objects move to far between two concurrent frames. If a person moves further than what is allowed between two frames the tracker will think that the previous object melted into the background and that a new object has appeared. The system updates the collaborating area with a very low frequency and the displacement could therefore be more than what the tracker tolerates. The tracker has a maximum length for allowed displacement of an object between two concurrent frames. This length could be increased but that would also increase the risk of losing an object between to frames. E.g., since the tracker looks for a similar object within this displacement distance it might match one person with his friend in the next frame and the person himself could then be in a state where the person is lost. Since it is not possible for an object to appear out of nowhere within a certain distance.

7.2 Performance of system without tracking

When tracking is disabled the system makes use of only background subtraction to find objects. This enables more than one person to be inside the collaborating area since each object then is independent of its location in the previous frame.

The problem one might face using this method is that objects are lost when they melt into the background. If an object is within any of the safety zones the system would simply forget it and the robot would operate in normal speed, higher than what is allowed. This could be avoided if the system would required that an object would exit each safety mode in correct order! The robot would also be possible to hit the object since it has no information of its location.

8 Conclusion and Discussion

8.0.1 Discussions

This system is based on a background segmentation to find moving objects in the scene which is used to extract points in the depth image. This is only one way of solving this, another more simple way of doing it would be to check at the depth image at once and extract information from it. For instance if the collaboration area would only consist of the floor and a robot objects would be found by thresholding the depth value in the image, that is removing what is floor in the image. This solution one might find more intuitive and simple but it does not take the movement of objects in the scene into account. Another disadvantage when having a background segmentation determining what to use in the image is that there are sometimes parts of objects that are misclassified as background. This problem makes the system more inaccurate and can make a distance between object and robot vary. In an area where there are objects, such as walls, visible for the camera one might prefer using a background segmentation since objects that are not moving will be removed and therefore the calculations needed will reduce significantly.

One option in the system is to turn off tracking. If it is guaranteed that no one will approach the robot and then stand still, the background segmentation will be sufficient.

It is remarkable how well the background segmentation works for the system compared to other similar systems. The reason is that the depth image, used in this system, do not have shadows, reflections or illumination differences in it. The absence of those visual artefacts makes the values of background pixels very constant. Constant background pixels will give a noise free foreground image.

In this project it has only been one kinect camera in use, this means that the position of the camera is of high importance since it needs to cover up the entire collaboration area. Unfortunately this was not possible to achieve and the camera setup for this project was therefore not perfect. The resulting camera position was not exactly over the robot, meaning that the system received a tilted view of the scene. A tilted view of the scene can result in objects which in fact are not connected to each other to be interpreted as connected, that is one object instead of two. In a more ideal case, it would be preferred to use more than one camera or have the camera placed exactly above the robot.

8.0.2 Uncertainty of calibration

There is an uncertainty in the accuracy of the calibration. While the computed relation between the known calibration pattern and the camera is based on very robust methods the static calibration between the robot and the pattern is manually measured. Even minor angular errors can give large discrepancy between the robots frame of reference and the cameras frame of reference. To mitigate this the calibration pattern was placed as flat as possible relative to the ground, since this angle is easier to verify than other angles (using e.g. a spirit level). This angle is not optimal from the perspective of detecting the pattern and therefore a rather large calibration pattern is used to ensure detection.

In a more optimal situation the pattern would be painted on something which is mechanically forced into a well known high precision static transform to the robot.

An example of such a system would be to use painted QR codes on different known spots with a large relative distance between them. Since QR-codes are designed to be easily detected with cameras and can convey an orientation as well as an identifier then the system could recalibrate as long as it can see at least one of them (though for the sake of robustness it probably should not unless it can see four of them). The feasibility of such a system is however out of the scope of this project.

8.0.3 Conclusion

The purpose of the project was to create a robust system which ensures a safe human and robot collaboration area. The system described in this report can not ensure this since there is no communication with the robot. As we discussed there is also other similar solutions to this project which can be used instead. However this project clearly shows that a system based on the same theory as this project can be used in industry to ensure the safety of both human and robots.

References

[1] OpenCV Background subtractor
`http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html?highlight=backgroundsubtractor#BackgroundSubtractorMOG2:publicBackgroundSubtractor`

[2] Cluster extraction PCL
`http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php`

[3] Greg James, Fast point in cylinder test
`http://www.flipcode.com/archives/Fast_Point-In-Cylinder_Test.shtml`

[4] Z. Zhang. A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000

[5] OpenCV camera calibration
`http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`

[6] Course homepage for TSBB15, found at URL: `http://www.cvl.isy.liu.se/education/undergraduate/tsbb15/3d-reconstruction-project`